A picture containing text, emblem, symbol, logo

Description automatically generatedElektrotehnički fakultet   
Univerziteta u Beogradu

Softversko inženjerstvo velikih baza podataka  
Domaći zadatak

Primena modela mašinskog učenja na problemu „Multi-class prediction of obesity risk“

Kandidat: Mentori:

Marko Brkić dr Miroslav Bojović  
3147/2023 ms Stefan Tubić

**Beograd, feb 2024.**

**SADRŽAJ**

[1. Uvod 3](#_Toc158150001)

[2. Analiza podataka 4](#_Toc158150002)

[2.1. Učitavanja korišćenih Pajton paketa 4](#_Toc158150003)

[2.2. Učitavanja zadatih podataka 4](#_Toc158150004)

[2.3. Inicijalni pregled podataka 5](#_Toc158150005)

[2.4. Analiza odlika i ciljne promenljive 6](#_Toc158150006)

[3. Pretprocesiranje podataka 9](#_Toc158150007)

[3.1. Binarno enkodovanje odlika 9](#_Toc158150008)

[3.2. Enkodovanje kategoričkih odlika 9](#_Toc158150009)

[3.3. Uvođenje novih odlika 9](#_Toc158150010)

[3.4. Korelacija odlika i ciljne promenljive 10](#_Toc158150011)

[4. Modeli mašinskog učenja 12](#_Toc158150012)

[4.1. Decision tree model 12](#_Toc158150013)

[4.2. Random Forest model 13](#_Toc158150014)

[4.3. XG Boost model 15](#_Toc158150015)

[4.4. Neuralna mreža 16](#_Toc158150016)

[4.5. CatBoost 17](#_Toc158150017)

[5. Zaključak 18](#_Toc158150018)

[5.1. Literatura 19](#_Toc158150019)

# Uvod

Gojaznost i kardiovaskularne bolesti predstavljaju dva najrasprostranjenija globalna zdravstvena problema, koji su poslednje vreme u porastu, i utiču na zdravlje i kvalitet života velikog procenta populacije. „Playground Series S4E2“ takmičenje na Kaggle platformi se zasniva na sintetičkom setu podataka koji je baziran na raznovrsnim podacima iz realnog života skupljenim od ljudi u Meksiku, Peruu i Kolumbiji, koji obuhvataju ljude svih uzrasta, navika u ishrani i drugih fizičkih karakteristika.

Cilj je odrediti nivo gojaznosti koji može biti *Underweight*, *Normal*, *Overweight*, *Obesity I*, *Obesity II* ili *Obesity III*.

Podaci koje dataset sadrži su:

* Gender – pol
* Age – uzrast
* Weight – kilaža
* Height – visina
* Family history with overweight – porodična istorija gojaznosti (istinitosna vrednost)
* FAVC – frekvencija konzumiranja visoko-kalorične hrane
* FCVC – frekvencija konzumiranja povrća
* CAEC – frekvencija broja malih obroka, tj. užina
* NCP – broj obroka na dan
* MTRANS – sredstvo transporta
* CALC – frekvencija konzumiranja alkohola
* TUE – vreme potrošeno na tehnološke uređaje
* FAF – frekvencija fizičke aktivnosti
* SCC – praćenje unosa kalorija (istinitosna vrednost)
* CH20 – broj dnevno popijenih litara vode
* SMOKE – istinitosna vrednost, da li je osoba pušač

# 2. Analiza podataka

## 2.1. Učitavanja korišćenih Pajton paketa

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.linear\_model import SGDClassifier

import xgboost as xgb

from catboost import CatBoostClassifier

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report, accuracy\_score

Uvezeni su česte biblioteke za analizu podataka i mašinsko učenje Pandas, NumPy, Matplotlib, Seaborn i SciKit-learn [1], kao i biblioteke koje sadrže mašinske modele koji su korišćeni u vežbi – XGBoost, CatBoost i TensorFlow koji je korišćen za neuralnu mrežu.

## 2.2. Učitavanja zadatih podataka

TEST\_DATASET = "test.csv"

TRAIN\_DATASET = "train.csv"

pd.set\_option('display.max\_columns', None)

# load data

df\_train = pd.read\_csv(TRAIN\_DATASET)

df\_test = pd.read\_csv(TEST\_DATASET)

## 2.3. Inicijalni pregled podataka

# examine data

print(df\_train.head())

print(df\_train.info())

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

## 2.4. Analiza odlika i ciljne promenljive

Osim ciljne promenljive, najkorisnije je detaljnije istražiti one koje nisu brojevnog tipa, kako bi programer odlučio kako će da ih enkoduje za mašinski model.

# analyze target variable

print(df\_train['NObeyesdad'].value\_counts())

plt.figure(figsize=(10, 6))

sns.countplot(data=df\_train, x='NObeyesdad')

plt.xticks(rotation=45)

plt.title('Analiza target varijable')

plt.show()

A graph of blue rectangular bars

Description automatically generated with medium confidence

Slika 1- Prikaz tipova gojaznosti u datasetu

Može se primetiti da je broj svih mogućih ishoda za gojaznost veoma balansiran, kao i da je najčešći „najgori“ tip gojaznosti. Ovakva raspodela ciljne promenljive pozitivno utiče na performanse našeg modela, koji će naučiti podjednako od svake klase.

Veoma interesantna činjenica koja je primećena tokom analiziranja podataka je da su u podacima za treniranje velika većina ispitanika tipa gojaznosti 3 žene, a tipa gojaznosti 2 muškarci, što može biti korisno za „post-processing“ tj. procesiranje posle provlačenja podataka kroz model.

plt.figure(figsize=(12, 6))

sns.countplot(x='NObeyesdad', hue='Gender', data=df\_train, palette='coolwarm', order = df\_train['NObeyesdad'].value\_counts().index)

plt.title('Gender Distribution in Each Weight Class', fontsize=16)

plt.xlabel('Obesity Level', fontsize=14)

plt.ylabel('Count', fontsize=14)

plt.xticks(rotation=45)

plt.show()

A graph of a bar chart

Description automatically generated with medium confidence

Slika 2 – Distribucija tipova gojaznosti po polu

# analyzing and encoding CALC, CAEC and MTRANS (mode of transport)

for df in [df\_train, df\_test]:

    calc\_distribution = df['CALC'].value\_counts(normalize=True) \* 100

    print(calc\_distribution)

    calc\_distribution.plot(kind='bar', title='Distribution of Alcohol Consumption')

    plt.ylabel('Percentage')

    plt.show()

    mtrans\_distribution = df['MTRANS'].value\_counts(normalize=True) \* 100

    print(mtrans\_distribution)

    mtrans\_distribution.plot(kind='bar', title='Distribution of Mode of Transportation')

    plt.ylabel('Percentage')

    plt.show()

    caec\_distribution = df['CAEC'].value\_counts(normalize=True) \* 100

    print(caec\_distribution)

    caec\_distribution.plot(kind='bar', title='Consumption of food between meals')

    plt.ylabel('Percentage')

    plt.show()

Analizirane su odlike CALC, CAEC i MTRANS kako bi se bolje razumeli podaci pre enkodovanja. Otkriveno je da za razliku od dataset-a za treniranje u kome za kolonu o konzumiranju alkohola postoje 3 vrednosti: “Sometimes”, “no” i “Frequently”, u test dataset-u postoji i vrednost “Always”. Ovo pokazuje značaj analize podataka za treniranje i za testiranje – odluka je doneta da se u test podacima “Always” redovi označe sa “Frequently” kako ne bi unosili konfuziju modelu.

A graph with blue squares

Description automatically generated

Slika 3 – Distribucija frekvencije konzumiranja alkohola

# 3. Pretprocesiranje podataka

## 3.1. Binarno enkodovanje odlika

Odlike čije su vrednosti istinosne vrednosti True i False ili koje imaju samo dve moguće vrednosti, možemo enkodovati za model mašinskog učenja veoma jednostavno nulama i jedinicama.

df\_train['Gender'] = df\_train['Gender'].map({'Male': 1, 'Female': 0})

df\_train['family\_history\_with\_overweight'] = df\_train['family\_history\_with\_overweight'].map({'yes': 1, 'no': 0})

df\_train['FAVC'] = df\_train['FAVC'].map({'yes': 1, 'no': 0})

df\_train['SMOKE'] = df\_train['SMOKE'].map({'yes': 1, 'no': 0})

df\_train['SCC'] = df\_train['SCC'].map({'yes': 1, 'no': 0})

## 3.2. Enkodovanje kategoričkih odlika

df\_train['CALC'] = df\_train['CALC'].map({'no': 0, 'Sometimes': 1, 'Frequently': 2, 'Always': 3})

df\_train['CAEC'] = df\_train['CAEC'].map({'no': 0, 'Sometimes': 1, 'Frequently': 2, 'Always': 3})

df\_train = pd.get\_dummies(df\_train, columns=['MTRANS'])

Odlikama CALC i CAEC dodeljene su brojne vrednosti po količini konzumiranja alkohola/frekvenciji manjih obroka – bitno je da vrednosti budu poređane ovako, jer ne bi imalo smisla da npr. 0 predstavlja odsustvo alkohola, a 1 visoku konzumaciju istog. Odlika MTRANS je podeljena „one-hot encodingom“ u 4 nove binarne odlike vezane za 4 sredstva prevoza u datasetu.

## 3.3. Uvođenje novih odlika

S obzirom na to da je set podataka fokusiran na gojaznost – ideja je bila da se doda odlika „BMI“ tojest body-mass index, koji je česti indikator gojaznosti u realnom životu. Da su podaci realni a ne sintetički, ova odlika sama po sebi bi verovatno mogla da uspešno predvidi kategoriju test podataka u 100% slučaja. Osim BMI-ja, isprobane su i odlike Diet Quality Score i Age Group, čije su ideje bile redom da se doda promenljiva koja spaja više odluka vezane za dijetu u jednu. One nisu donele poboljšanje skora pa nisu uključene u krajnje rešenje. Glavni razlog za neuspeh Age Group-a je činjenica da je dataset bio zasićen mladim odraslim ljudima, pa je većina ljudi upadalo u jednu grupu uzrasta.

# add BMI feature

df\_train['BMI'] = df\_train['Weight'] / (df\_train['Height'] \*\* 2)

# add  Diet Quality Score feature

w\_fcvc = 1  # Frequency of consumption of vegetables

w\_favc = -1  # Frequent consumption of high caloric food

w\_ch2o = 1  # Consumption of water daily

w\_calc = -1  # Consumption of alcohol

w\_caec = 0.5  # Consumption of food between meals

w\_faf = 1  # Physical activity frequency

df\_train['DQS'] = df\_train['FCVC'] + \

                  (1 - df\_train['FAVC']) + \

                  df\_train['CH2O'] + \

                  (3 - df\_train['CALC']) + \

                  (3 - df\_train['CAEC']) + \

                  df\_train['FAF']

# use age groups instead of age

bins = [-np.inf, 17, 29, 59, np.inf]

labels = [0, 1, 2, 3] # children, young adults, adults, seniors

df\_train['Age'] = pd.cut(df\_train['Age'], bins=bins, labels=labels).astype(int)

print(df\_train['Age'].value\_counts())

print(df\_train.head())

## 3.4. Korelacija odlika i ciljne promenljive

Još korisnih informacija za programera može dati korelacija odlika i ciljne promenljive – tj. informacija kolika svaka odlika utiče na ishod. Ukoliko neka odlika utiče veoma malo, treba razmisliti o uklanjanju te kolone. U slučaju Kaggle takmičenja S4 E2, odlučeno je da se izbace kolone NCP i SMOKE, što je dovelo do poboljšanja učenja modela. Razlog za ovo je što činjenica da li neko puši i broj obroka na dan zapravo ne utiču puno na gojaznost, što je možda nekim ljudima čudan zaključak ali rezultati pokazuju drugačije. Naravno, i id kolona se izbacuje iz dataseta pre treniranja modela.

A graph of a number of numbers

Description automatically generated with medium confidence

Slika 4 – Korelacija odlika i ciljne promenljive

# remove SMOKE, NCP and id feature due to low correlation

df\_train = df\_train.drop('SMOKE', axis=1)

df\_train = df\_train.drop('NCP', axis=1)

df\_train = df\_train.drop('id', axis=1)

# 4. Modeli mašinskog učenja

Nakon uspešnog pretprocesiranja podataka, potrebno je opredeliti se za model mašinskog učenja koji daje najbolje rezultate. Iz ovog razloga nekoliko modela se treniraju nad podacima, i na osnovu standardnih metrika se procenjuje koji je najbolji. Preciznije, ova faza se izvršava naizmenično sa trećem, kako bi se razumelo da li neke od obrada podataka pozitivno ili negativno utiču na rezultat.

Metrike koje su accuracy (tačnost) i F1 skor, koji predstavlja harmonijsku sredinu preciznosti i opoziva (Recall). Takođe korisno je pregledati matricu konfuzije, koja može vizuelno pomoći u razumevanju u kojim predikcijama je model pogrešio.

## 4.1. Decision tree model

# Decision Tree

dt\_classifier = DecisionTreeClassifier(random\_state=1)

dt\_classifier.fit(X\_train, y\_train)

y\_pred = dt\_classifier.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy for Decision Tree: {accuracy\*100:.2f}%")

f1 = f1\_score(y\_test, y\_pred, average='weighted')

print(f"F1 Score for Decision Tree: {f1:.2f}")

cm = confusion\_matrix(y\_test, y\_pred)

cm\_df = pd.DataFrame(cm, index=class\_labels, columns=class\_labels)

sns.heatmap(cm\_df, annot=True, fmt='g', cmap='Blues', cbar=False)

plt.title('Confusion Matrix')

plt.ylabel('Actual Labels')

plt.xlabel('Predicted Labels')

plt.show()

Decision tree je popularan i jednostavan model mašinskog učenja koji se može koristiti i za klasifikaciju i za regresiju. Funkcioniše tako što raspodeli naše podatke u više manjih setova podataka za koje napravi drvo odluka pomoću kog može zaključiti ciljnu klasu. Najveći problem ovog algoritma je što male promene u podacima mogu da naprave velike promene u drvetu – zbog ovoga se koristi Random Forest algoritam koji prave “šumu” drveća. [2]

Tačnost ovog modela bila je 83.60%, dok je F1 skor bio 0.84 što pokazuje da je ovaj model uz pretprocesiranje dao sasvim solidan rezultat, i očekuje se još bolji rezultat sa naprednijim modelima. Na matrici konfuzije se jasno može videti da je naš rezultat dobar, jer je većina dataseta na dijagonali, dok se može primetiti da je najčešće bilo mešanje tipa gojaznosti 2 i 3.

A screenshot of a computer

Description automatically generated

Slika 5 – Matrica konfuzije modela Decision Tree

## 4.2. Random Forest model

Kao što je već napomenuto, prednost random forest modela je to što kreira brojna drveća sa nasumičnim delovima podataka. [3] Ovo omogućava da se svako drvo trenira na različitom setu podataka i time mnogo manje zavisi od ulaznih podataka.

# Random Forest classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_classifier.fit(X\_train, y\_train)

y\_pred = rf\_classifier.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy for Random Forest: {accuracy\*100:.2f}%")

f1 = f1\_score(y\_test, y\_pred, average='weighted')

print(f"F1 Score for Decision Tree: {f1:.2f}")

cm = confusion\_matrix(y\_test, y\_pred)

Tačnost ovog modela je 89.72% dok je F1 skor 0.9, što predstavlja poboljšanje u odnosu na Decision Tree model što je bilo očekivano. Matrica konfuzije izgleda veoma slično, s tim da je broj loše predviđenih klasa opao.

A graph with numbers and a number in blue squares

Description automatically generated

Slika 6 – Matrica konfuzije modela Random Forest

## 4.3. XG Boost model

XGBoost, tj. eXtreme Gradient Boosting, je napredni algoritam koji se zasniva na konstruisanju novih modela, koji pokušavaju da predvide greške prethodnih modela i da se poboljšaju, pa na kraju spajaju svoje predikcije.

target\_mapping\_xg\_boost = {

    'Insufficient\_Weight': 0,

    'Normal\_Weight': 1,

    'Overweight\_Level\_I': 2,

    'Overweight\_Level\_II': 3,

    'Obesity\_Type\_I': 4,

    'Obesity\_Type\_II': 5,

    'Obesity\_Type\_III': 6

}

y\_train\_encoded = y\_train.map(target\_mapping\_xg\_boost)

y\_test\_encoded = y\_test.map(target\_mapping\_xg\_boost)

xgb\_classifier = xgb.XGBClassifier(use\_label\_encoder=False, eval\_metric='mlogloss') #, n\_estimators=500, max\_depth=8, learning\_rate=0.01, subsample=0.8, colsample\_bytree=0.8, gamma=1, reg\_alpha=0.01, reg\_lambda=1

xgb\_classifier.fit(X\_train, y\_train\_encoded)

y\_pred = xgb\_classifier.predict(X\_test)

accuracy = accuracy\_score(y\_test\_encoded, y\_pred)

print(f"Accuracy for XGBoost: {accuracy\*100:.2f}%")

f1 = f1\_score(y\_test, y\_pred, average='weighted')

print(f"F1 Score for Decision Tree: {f1:.2f}")

cm = confusion\_matrix(y\_test, y\_pred)

Korak procesiranja koji se primećuju na početku koda potreban je jer XGBoost ume da radi samo sa numeričkim ciljnim promenljivama. Preciznost modela je 90.13%, dok je F1 skor 0.9. Na osnovu matrice konfuzije može se primetiti da je ovaj model savršeno raspodelio klase gojaznosti nivoa 2 i 3, što je značajno poboljšanje u odnosu na prošli model. Ova činjenica pokazuje da je ovaj model bio bolji u generalizovanju svog znanja, i nije se previše obzirao na trening dataset (izbegao je overfitting). Važno je napomenuti da je ovaj model najbolje funkcionisao na Kaggle platformi, sa uspešnosti od 91.04%.

A screenshot of a computer

Description automatically generated

Slika 7 – Matrica konfuzije modela XGBoost

## 4.4. Neuralna mreža

# Neural Network

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Normalization of attributes

y\_train\_categorical = tf.keras.utils.to\_categorical(y\_train\_encoded)

y\_test\_categorical = tf.keras.utils.to\_categorical(y\_test\_encoded)

model = Sequential()

model.add(Dense(128, input\_dim=X\_train\_scaled.shape[1], activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dense(y\_train\_categorical.shape[1], activation='softmax'))  # Use softmax for multi-class classification

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(X\_train\_scaled, y\_train\_categorical, validation\_split=0.2, epochs=20, batch\_size=32)

loss, accuracy = model.evaluate(X\_test\_scaled, y\_test\_categorical)

print(f"Accuracy for NN: {accuracy\*100:.2f}%")

1. Skaliranje u kodu je korišćeno kako bi se normalizovale sve odlike od 0 do 1, što omogućava da sve jednako doprinesu treniranju [4]. Neuralna mreža sadrži 128 neurona u prvom sloju i 64 u skrivenom, dok je, logično, izlazni sloj sastavljen od onoliko neurona koliko ima klasa. Tačnost neuralne mreže bila je 0.88 što je lošije od prethodnog modela, i može se zaključiti da su one veoma osetljive na promene hiperparametara, i da bi se promenom njih mogli postići bolji rezultati.

## 4.5. CatBoost

CatBoost tj. Categorical Boosting je model veoma sličan XGBoost-u, s tim da ima prednost da može direktno da barata sa kategoričkim odlikama.

# CatBoost

catboost\_clf = CatBoostClassifier(logging\_level='Silent',iterations=1000,

    learning\_rate=0.1,

    depth=6,eval\_metric='Accuracy',

    random\_seed=1)

catboost\_clf.fit(X\_train, y\_train)

y\_pred = catboost\_clf.predict(X\_test)

y\_pred = y\_pred.ravel()

Lokalno, ovaj model se najbolje pokazao sa tačnošću od 90.68% i F1 skorom od 0.91. Matrica konfuzije pokazuje da ovaj model, iako nije dobro raspodelio tipove gojaznosti 2 i 3, to je nadoknadio na drugim klasama.

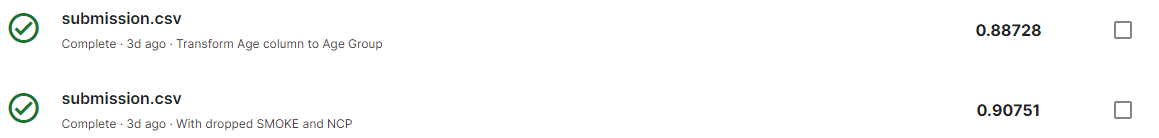
A graph with numbers and squares

Description automatically generated

Slika 8 – Matrica konfuzije modela CatBoost

# 5. Zaključak

Zaključuje se da su dva najbolja modela XGBoost i CatBoost, koji se najbolje pokazuju redom na Kaggle-u i XGBoost-u. Takođe primećuje se da je potrebno validirati pretpostavke napravljene tokom pretprocesiranja podataka, jer postoji šansa da je neka od njih pogrešna i zapravo dovodi do lošije predikcije. To se u ovom slučaju desilo pri pretpostavci da bi Age Group odlika donela bolje rezultate.



Slika 9 – Neuspešna odlika Age Group

Da bi se rešenje poslalo na Kaggle, potpuno je odraditi isto pretprocesiranje na test podacima i zatim napraviti submission.csv fajl kojim Kaggle ocenjuje rešenje.

y\_test\_pred = xgb\_classifier.predict(df\_test)

y\_test\_pred = y\_test\_pred.ravel()

reverse\_target\_mapping = {v: k for k, v in target\_mapping\_xg\_boost.items()}

y\_test\_pred\_names = [reverse\_target\_mapping[label] for label in y\_test\_pred]

submission = pd.DataFrame({

    'id': df\_test\_ids,

    'NObeyesdad': y\_test\_pred\_names

})

submission.to\_csv('submission.csv', index=False)

Buduće poboljšanje modela moglo bi da obuhvata kombinovanje najboljih performansi XGBoost i CatBoost modela, pošto XGBoost najbolje klasifikuje česte tipove gojaznosti 2 i 3, dok CatBoost najbolje izvršava klasifikaciju drugih klasi. Takođe, uz detaljno konfugirusanje hiperparametra svih modela, a najviše neuralnih mreža, rezultati bi mogli biti poboljšani. [5]

## 5.1. Literatura

Za navođenje referenci korišćen je IEEE format.

|  |  |
| --- | --- |
| [1] | „Wikipedia - scikit-learn,“ [Na mreži]. Available: https://en.wikipedia.org/wiki/Scikit-learn. |
| [2] | E. F. M. A. H. Ian H. Witten, Data mining - Practical Machine Learning Tools and Techniques. |
| [3] | J.-M. P. Robin Genuer, Random Forests with R, 2018. |
| [4] | „geeksforkeegs.org - Data Preprocessing in Data Mining,“ [Na mreži]. Available: https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/. |
| [5] | C. C. Aggarwal, Neural Networks and Deep Learning, 2018. |